Introduction to the Atari TOS Developer's Kit

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94089
(408) 745-2000

February 28, 1992

1. Introduction

Thank you for your interest and support of the Atari TOS-based series of computers. Enclosed is the documentation and software package you have ordered. Read this section first to learn how to quickly organize your documentation and get on the road to ST product development.

A bit of warning is in order at this point. This document, and the other documents in this developer's kit, assumes that the reader has some experience using the Atari system, even if they are not knowledgable about programming it.

It is recommended you become familiar with the operation of the GEM Desktop and at least a few application programs before attempting any serious work with the developer's kit.

Additionally, most of the documentation in the developer's kit is intended as a reference, not a tutorial. If you aren't a programmer already, the developer's kit by itself won't make you into one. There are numerous excellent books available about the C programming language and 68000-family assembly language that you should obtain.

After you have installed your developer's kit, please see the **Basic Programming Guidelines** section at the end of this document.

2. Online Services Information

Atari provides technical support through the special ATARI.RSC Atari Developers' Roundtable section on the GEnie Telecommunications network.

To access the Roundtable, type ATARI.RSC at any GEnie prompt. ATARI.RSC is a restricted area, and only registered Atari developers are allowed. The first time you try to get in, the system won't know about you yet. You have to try to enter ATARI.RSC at least once so that the Atari developer support personnel will be able to grant you access.

A message will appear instructing you where to send EMAIL to gain access. You will be asked to input a developer identification number. If you don't know yours, leave your company name and the name of the primary contact.

Once you have access to ATARI.RSC, you can get technical support from Atari as well as from other Atari developers. You will also have access to restricted data libraries which have utilities and example programs.

The telephone numbers and EMAIL addresses for Atari's Developer Support staff can be found in the ATARI.RSC developer newsletter. Simply locate the most recent of the back issues included in your developer's kit.

3. Documentation In This Package

The documentation may seem overwhelming at first, but it is easy to get started. All pages are pre-punched for the three-ring binders. The sections you will need the most are:

The HitchHiker's Guide to the BIOS
Atari GEMDOS Reference Manual
GEM Programmer's Guide Volume 1 – VDI
GEM Programmer's Guide Volume 2 – AES
Programmer's Guide to FSMGDOS

To start programming in Alcyon C, the first files you should read are on the MicroEMACS disk. They will instruct you on the use of the MicroEMACS text editor.

The document you are now reading primarily contains information about installing the developer's kit software onto your hard disk, then compiling and linking your first program. Later you may refer to other documents in the package for detail on changing parameters of the compiler, linker, and other utilities.

See the enclosed **Contents of Atari TOS Developer's Kit** list to be sure that your kit is complete. There are copies of newsletter back-issues to bring you up-to-date. These may also contain corrections to other pieces of documentation in the developer's kit.

Programming references include the following:

GEM Programmer's Guide Volume 1 – VDI
GEM Programmer's Guide Volume 2 – AES
Programmer's Guide to FSMGDOS
HitchHikers's Guide to the BIOS
Atari GEMDOS Reference Manual
Pexec Cookbook
AHDI 3.00 Release Notes (includes later versions)

Manuals on the development tools include:

C Language Programming Guide
ALN Linker manual
MADMAC assembler manual
DB debugger manual
Atari CHKDISK3 manual
Resource Construction Set

The following are more advanced documents for the developer who wants to access the lowest level of the operating system:

Intelligent Keyboard Protocol (IKBD)

Hardware documentation is included for those building add-on products:

Engineering Hardware Specifications
Chip specifications:
6850 ACIA
MK68901 Multi-Function Peripheral
The Sound Chip
Programmable Sound Generator Data
AY-3-8910 Programmable Sound Generator
WD 1770 Floppy Disk Controller
Atari Monitor Summary Specifications
128K ROM cartridge schematic
Atari ST Bit-Block Transfer Processor Blitter Chip

There are also supplements to cover newly released enhancements and new machines:

Rainbow TOS Release Notes
STE Developer's Addendum
STE TOS Release Notes
TT030 TOS Release Notes
STBook Expansion Bus Specification

Finally, additional information can be ordered by those who have special development requirements. These documents may involve additional developer's agreements or additional cost:

GDOS Developers Kit
CDAR CD ROM Developers Kit
Field Service Manuals, including schematics
Atari SFP004 Floating Point Coprocessor
Mega ST Series Internal Expansion Bus
Atari ACSI/DMA Integration Guide
Still Another Line-A Document (SALAD)

Besides the documentation in this package, there are also additional documents which should be obtained directly from the manufacturer/supplier

MIDI Specification:
IMA - The International MIDI Association
11857 Hartsook St.
North Hollywood, CA 91607
(818) 505-8964

NCR 5380
SCSI Controller Chip Specification:
NCR Microelectronics
1635 Aeroplaza Drive
Colorado Springs, CO 80916
(719) 596-5612

VMEbus Specification:
VITA
10229 N. Scottsdale Rd., Suite B
Scottsdale, AZ, 85253
(602) 951-8866

Zilog 85C30
Serial Communications Controller Specification:
Zilog
210 Hacienda Ave.
Campbell, CA, 95008-6609
(408) 370-8000

4. Disks in this Package

The ST development package includes several disks. These disks will enable a developer to compile and link "C" or assembly language applications or desk accessories for the Atari ST/TT series. They also provide samples of programming techniques.

1) Alcyon C Compiler
Linker

2) MicroEMACS
Resource Construction Set
Source Code Examples

3) Programming Utilities
DB Debugger
MADMAC Assembler

4) Atari Hard Disk Utilities
System Utilities
STE Programming Examples

5) Demonstrator
eXtensible Control Panel

WARNING: It is recommended that the developer make BACKUP COPIES of these disks and put the originals away in a safe place. Use the backups ONLY!

The list of files, programs, and other information that is included on these disks is in the **Contents of Atari TOS Developer's Kit** document.

5. Hardware Requirements

The minimum system requirement for the Atari TOS Developer's Kit is a 1 megabyte TOS-based system with a hard disk drive and at least one double-sided floppy disk drive. A 4 megabyte system is strongly recommended, as this will better facilitate the use of the RAM disk program included with the kit.

It is further recommended that the serious developer test their software in all possible video modes. Also remember that there are older revisions of the operating system, as well as international versions. Systems with various memory sizes should be used in testing a program. The developer should follow the proper programming protocols in order to allow their programs to function under any hardware configuration. See the section **Basic Programming Guidelines** at the end of this document.

At the time of this writing (Nov. 6, 1991) TOS-based machines include the following: 520ST, 520STFM, 520STE, 1040SFMT, 1040STE, Mega ST, Mega STE, TT030, STacy, STBOOK.

6. Installing the Developer's Kit

Follow the instructions below to install the contents of the developer's kit diskettes onto your hard disk drive. Approximately 3.5 megabytes of free disk space is required for installation of all the files. The instructions assume drive A on your system is a double-sided disk drive. If necessary, use drive B instead.

1) Create a folder on the desired hard disk partition to contain all of the developer's kit information. We shall refer to this folder as the DEVKIT folder (even if you actually use a different name).

2) Insert Disk 1 into drive A. Copy the DEV folder from the floppy to the DEVKIT folder.

3) Insert Disk 2 into drive A. Copy the EMACS folder, the EXAMPLES folder, and the RCS folder to the DEVKIT folder.

4) Copy the ME.TTP program from the EMACS folder on drive A into the DEV folder within the DEVKIT folder on the hard disk. (After you have learned to use the MicroEMACS text editor, you can delete the EMACS folder from your hard disk.)

5) Insert Disk 3 into drive A. Copy the DB folder and MADMAC folder to the DEVKIT folder.

6) Copy the contents of the UTILITY folder on drive A into the DEV folder on the hard disk.

7) Insert Disk 4 into drive A. Copy the HDX folder, the STE folder, and the SYSTEM folder to the DEVKIT folder.

8) Insert Disk 5 into drive A. Copy the DEMONSTR.TOR folder and XCONTROL folder to the DEVKIT folder.

9) Your developer's kit is now installed.

7. Compiling a program with Alcyon C

The compiling and linking of a program with Alcyon C is a very simple process. The following is an example on how to compile and run the sample APSKEL.C program supplied in the developer's kit.

Note: If you have a C compiler or assembler (or other programming language) other than that included with the developer's kit, and you would prefer to use it instead of Alcyon C, please refer to the documentation that came with it for information on compiling or assembling a program. However, please remember that the example programs in the developers kit are designed to be used with the tools from the developer's kit. Some modifications may be required if you want to use a different compiler or assembler.

1) Open a window on the GEM Desktop for the DEV folder created when you installed the developer's kit software, as described in section 6 of this document.

2) Select the BATCH.TTP program and open it.

3) When the open application dialog box appears, type "C APSKEL" and select "OK". The batch program will execute the C compiler to begin compiling the APSKEL.C program. You will see some status messages on the screen as the compile proceeds.

   The compiler is finished when the screen shows a message asking for a carriage return. Hit the [Return] key to return to the desktop.

4) Double-click on the BATCH.TTP program again. This time, type "LINKAP APSKEL" in the open application dialog box and select "OK".

   The batch program will execute the linker to link the proper object files and libraries together. The linker is finished when the screen shows a message asking for a carriage return. Hit the [Return] key to return to the desktop.

5) Now just double-click on the file APSKEL.PRG that has appeared in the desktop window.

   Within the APSKEL program, the window's move bar, size button, close button, and full button are all active. The close button returns you to the desktop.

To compile your own program would be very similar, except that you would use the name of your program instead of "APSKEL".

Warning! Remember not to include the ".C" portion of the filename in the Open Application dialog, or your original source code file will be deleted! For example, enter "APSKEL" into the dialog, not "APSKEL.C"

7.1 Compiling a Desk Accessory

To compile and link the desk accessory ACSKEL.C program, follow the same procedure as listed above, but with the following changes: use ACSKEL wherever you would have used APSKEL, and instead of LINKAP in step 4, use LINKACC.

1) The final step is to transfer the ACSKEL.ACC accessory you have just created to a disk in drive A and reboot the system from the floppy. Rebooting the system installs the accessory under the **Desk** menu.

   It is recommended that desk accessories be tested from floppy disk. If there are problems with the desk accessory, it may crash before you have an opportunity to get to the desktop to disable it, which would be quite annoying to work around if you are booting from your hard disk.

   The boot program for most hard drives for the Atari can be disabled by holding down some combination of the Control, Alternate, and Shift keys during the boot process. Consult your manual for more information.

2) To run, move the mouse to DESK on the menu bar and click on "Sample Accessory". If you resize the window you will see disk icons and other windows on the desktop. Selecting the close box will remove the desk accessory.

If you view the contents of C.BAT and LINKAP.BAT, you will be able to see the proper order of instructions necessary to compile or link a program file. For more information on compiling and linking, please consult the Atari ST developer's kit manuals.

The two remaining link batch files, LINKIO and CLINK, are used for creating either TOS or TTP type applications with GEMDOS and the C runtime libraries, respectively. No examples are provided with this discussion.

The batch files included with the developer's kit assume that the programmer's source code is located in the DEV folder (that is, the same folder as the compiler and linker). To have your source code in a different folder from the compiler and linker, you can create custom batch files which point to your desired folders.

8. Include / Link Information

The following information represents the proper C header files which must be included to correctly call TOS functions, as well as the libraries necessary to link in the proper functions into your program.

System	Include	Link With...
VDI	define.h vdibind.h	VDIBIND library
AES	define.h aesbind.h obdefs.h	AESBIND library
GEMDOS BIOS XBIOS	define.h osbind.h	OSBIND.O object module
"C" library	stdio.h	GEMLIB library
Floating Point math	math.h	LIBF library

The LIBF math library is also required when certain functions of the GEMLIB C library are used, and should be linked with your program when using any C library function which takes a floating point value as a parameter or which returns a floating point value.

9. Startup Module & Stack Size

There are several startup modules included with the developer's kit. These are intended to be used as the very first object module when linking a program together. They contain special initialization code and data tables used by the GEM VDI & AES libraries and certain parts of the C runtime library.

The source code for APSTART, ACCSTART, and GEMSTART is included with the developer's kit and may be customized and re-assembled for your own needs.

GEMSTART.O – For applications which use the C runtime library. Please note that many of the functions in the C runtime library GEMLIB can be used without the initialization code included in GEMSTART.O.

APSTART.O – For applications which do not use the C runtime library.

ACCSTART.O – For desk accessories. Can be used with most standard C library calls by linking with GEMLIB and LIBF.

For applications, one would normally include use the GEMSTART startup module. For Desk Accessories, one would use the ACCSTART module.

GEMSTART is a greatly enhanced version of APSTART which includes additional initialization code for certain functions in the C runtime library.

The Stack size for applications and desk accessories has been set at 1K. If this is not enough for your program, make the appropriate changes to the startup module you are using and re-assemble it.

10. Developer Kit Utilities

There are several utility programs included with the developer's kit which are not described in the other documentation.

BATCH.TTP is a batch processing program which runs through a batch file and executes the programs named within the batch file with the specified parameters.

For example, to execute the C.BAT file from the GEM Desktop, you would double-click on BATCH.TTP and when the Open Application dialog box appeared, you would enter "C myfile".

When BATCH.TTP is given the line "C myfile" it automatically adds ".BAT" to the first part, and then reads the file C.BAT, as shown below:

```
cp68 %1.c %1.i
c068 %1.i %1.1 %1.2 %1.3 -f
rm %1.i
c168 %1.1 %1.2 %1.s
rm %1.1
rm %1.2
as68 -l -u %1.s
rm %1.s
wait.prg
```

The "%1" you see in several places is a placeholder for the first parameter you will enter into the dialog. When you enter "C myfile", the BATCH.TTP takes the first parameter after the "C" and uses it everywhere it finds "%1" in the C.BAT file. So that the first line becomes:

```
cp68 myfile.c myfile.i
```

If there had been more parameters after "myfile" then it would have looked for "%2" and replaced it, and so on.

BATCH.TTP then runs the specified programs and passes along the parameters. For example, it would run CP68.PRG and give it a commandline containing "myfile.c myfile.i". When that program was finished, it would go to the next file and execute it, until it got to the end of the C.BAT file.

If you want to use a commandline shell instead of the GEM Desktop (such as the COMMAND.TOS program included with the developer's kit), then you would run BATCH.TTP like this:

```
batch c myfile
```

There are several more sophisticated commandline shells available, most of which include built-in batch file processing. These may not support the same batch files as BATCH.TTP without some modification.

There are several batch files included with the developer's kit which you may examine to learn how to create your own.

RM.PRG is a program which deletes the file specified in the commandline. When running BATCH.TTP to compile files, RM.PRG is typically callled by the batch file to delete temporary files created by the compiler or assembler after they are no longer needed.

WAIT.PRG is a program which prints a message and waits for a key to be pressed. WAIT.PRG is used in batch files so that the programmer has time to view screen output, such as error messages, before they are scrolled off-screen or disappear upon returning to the GEM Desktop.

11. Global Variable Names To Avoid

The startup modules and libraries included in the developer's kit use several global variable names which should be avoided in your program. The linker always

takes the most recent definition of anything as being the "real" definition. If you had an array in your program named max[ ] and you linked with the VDIBIND library, the linker would confuse the array in your program with the max( ) function included in VDIBIND.

Below is a list of global variable names which are used by the various startup modules and libraries, and which should be avoided by your programs.

ACCSTART.O, APSTART.O startup modules

```
crystal          ctrl_cnts
```

GEMSTART.O startup modules

```
_main            _exit
_start           _cpmrv
_base_sw         _sovf_break
_BDOS            _pname
_tname           _lname
_xeof            blkfillindex
strchr
```

AESBIND library

```
c                control
global           int_in
int_out          addr_in
addr_out         gl_apid
ad_c
```

VDIBIND library

```
i_ptr            i_ptr2
m_lptr2          mul_ptr
MUL_DIV          smul_div
umul_div         gsx1
SMUL_DIV         iioff
gsx2             UMUL_DIV
iooff            pioff
pooff            vdi
vec_len          max
```

OSBIND.O object module

```
gemdos           bios
xbios
```

12. Basic Programming Guidelines

Below is a set of basic guidelines for programming practices meant to insure that your programs are compatible across current and future machines. Developers are urged to strive for compatibility as much as possible. While this list is not complete by any means, it will hopefully point you in the right direction.

Re-read this section after you have become familiar with the documentation in the developer's kit. You may not immediately understand the references and reasons behind the guidelines, but they will become clearer as you learn more about the system.

1) Read all of the documentation included in the developer's kit before starting to program. It's a lot of reading, but it will save you from making some hard to find mistakes later on down the road.

2) Don't make assumptions about the screen. There will be new video modes in new machines and add-on video boards you didn't know about when you wrote your program. If you make assumptions about the display hardware, then your program will probably fail when something new comes along.

   The operating system provides information about the display in use. This is accomplished by opening a GEM VDI Workstation and examining the returned information. Additional information is available through GEM VDI inquire functions.

   Some basic rules and some of the more common assumptions to avoid are given below.

   a) A certain screen resolution does not mean a certain number of colors on screen at once, or vice versa.

   b) Don't assume the color palette is a certain number of colors because you're on a certain type of machine. The machine may have an add-on video board which provides more colors.

   c) Many early programs written for the ST made assumptions about the screen resolution by looking at the hardware video shifter mode through the Getrez() XBIOS function, which ultimately has nothing to do with the screen resolution.

   d) If the information is available from GEM VDI, then get it from GEM VDI, not someplace else, including Line-A. GEM VDI will always have the last word.

The use of Line-A is strongly discouraged. Documentation is still provided in the developer's kit about Line-A, but it should be avoided if at all possible. Most applications have no reason to use it. Contact Atari developer technical support for more information if required.

3) Don't assume only certain machines have certain hardware or software features.

   The Cookie Jar is an operating system feature that allows programs to obtain information about the system they are running on, such as if DMA stereo sound is available. The Cookie Jar should be used wherever possible when a program needs to know if a certain hardware or software feature is available. Use the machine type ONLY for detecting features which don't have a cookie of their own. The Cookie Jar is discussed in the STE TOS Release Notes.

4) Learn how to program the GEM AES before you learn to program the GEM VDI. The heart of a GEM program lies in the menus, windows, and dialogs through which the user must operate the program. These are all the responsibility of GEM AES. The sooner you learn to utilize GEM AES, the more polished and professional your programs will appear.

   Examples of existing applications which make good use of the GEM AES include Migraph's Touch-Up (with its pop-up menus and palettes), Gribnif/Strata's STalker and STeno (all-around good examples), The Resource Construction Set (included with the developer's kit), and the GEM Desktop.

   If you would like an idea of how to do something and you cannot find an existing example, please contact Atari developer technical support.

5) Don't do something different just for the sake of being different.

   When in doubt about how something should work or look, examine how the GEM Desktop looks and how it does things. If your program has something similar, it should follow the example of the GEM Desktop.

   When creating the user interface for your program, keep in mind that simple is better. In almost all cases, the simpler something is, the easier it is to learn, and the easier it is to use. Don't make extra steps for the user if you can avoid it.

6) Replacing, ignoring, or working around the resources of the operating system should be avoided.

   For example, there is a standard system file selector for a reason. A GEM program which doesn't use a standard file selector is inconsistent with the interface of standard applications. At least make it a user preference option in the application.

   As another example, some programs use a proprietary font format for their text capabilities, rather than using GEM's font and text functions. As a result, these programs did not work with the outline font capabilties of FSMGDOS when it became available. On the other hand, many older programs which did use GEM's font and text functions work just fine with FSMGDOS outline fonts. If a program wants to do something like support its own proprietary font format, it should be done in addition to supporting the standard GEM features, not instead of them.

7) Be familiar with the various update documents such as the **Rainbow TOS Release Notes** and **TT030 TOS Release Notes.** These will contain additions, corrections, and clarifications to other documents in the developer's kit.

8) Watch the ATARI.RSC developer newsletter for any documentation corrections or updates.

9) If you can use a vector to get a memory address, do it. Don't assume that something will always be at the same place in all machines and all versions of the operating system unless it is a documented memory address.

   For example, the operating system itself is in a different place in newer machines than with older machines. If an application needs to access the OS header to get a version number or country code, for example, then the application should not assume that the OS ROM starts at a certain memory location. Instead, it should get the address of the OS header from the appropriate system variable.

10) Do not use or rely upon undocumented variables in memory or *reserved* variables or parameters in a function call or data structure.. They **will** move or change when a new operating system revision is introduced.